Lab 5

Reg. No: 19BCE1209

Name: Gautam Sanjay Wadhwani

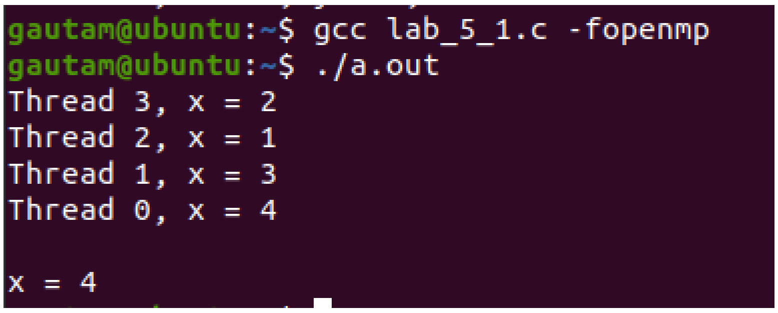
Course: CSE4001 Parallel and Distributed Computing

Q1. Example Program -- Critical

Code:

```
#include <stdio.h>
#include <omp.h>
int main()
{
int x = 0;
#pragma omp parallel
{
#pragma omp critical
{
x = x + 1;
}
printf("Thread %d, x = %d\n", omp_get_thread_num(), x);
}
printf("\nx = %d\n", x);
return 0;
}
```

Output:



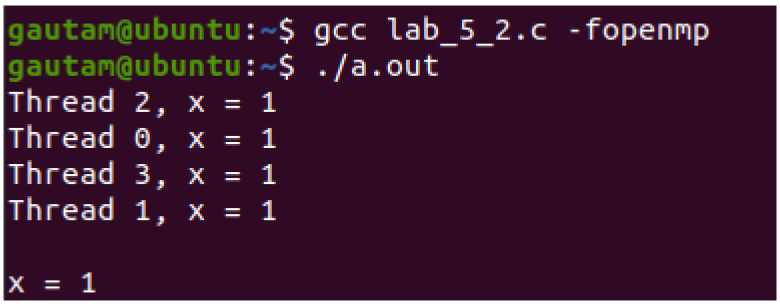
```
gautam@ubuntu:~$ gcc lab_5_1.c -fopenmp
gautam@ubuntu:~$ ./a.out
Thread 3, x = 2
Thread 2, x = 1
Thread 1, x = 3
Thread 0, x = 4

x = 4
```

Q2. Example Program -- Single

Code:

```
#include <stdio.h>

#include <omp.h>

int main()

{

int x = 0;

#pragma omp parallel

{

#pragma omp single

{

x = x + 1;

}

printf("Thread %d, x = %d\n", omp_get_thread_num(), x);

}

printf("\nx = %d\n", x);

return 0;

}
```

Output:



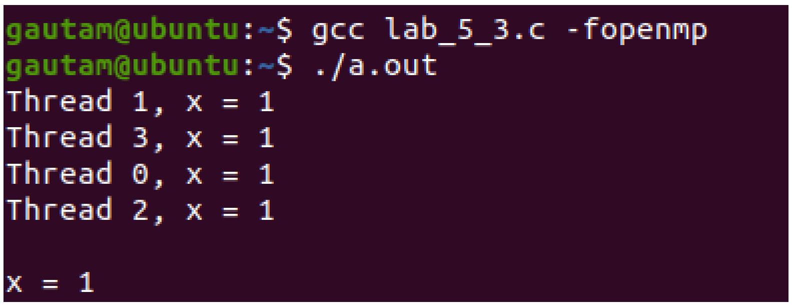
```
gautam@ubuntu:~$ gcc lab_5_2.c -fopenmp
gautam@ubuntu:~$ ./a.out
Thread 2, x = 1
Thread 0, x = 1
Thread 3, x = 1
Thread 1, x = 1

x = 1
```

Q3. Example Program -- Master

Code:

```
#include <stdio.h>

#include <omp.h>

int main()

{

int x = 0;

#pragma omp parallel

{

#pragma omp master

{

x = x + 1;

}

printf("Thread %d, x = %d\n", omp_get_thread_num(), x);

}

printf("\nx = %d\n", x);

return 0;

}
```

Output:



```
gautam@ubuntu:~$ gcc lab_5_3.c -fopenmp
gautam@ubuntu:~$ ./a.out
Thread 1, x = 1
Thread 3, x = 1
Thread 0, x = 1
Thread 2, x = 1

x = 1
```

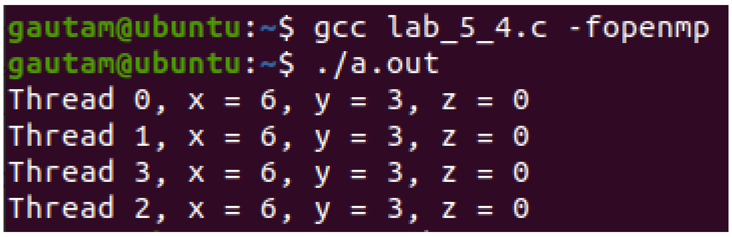
Q4. Example program with x=x+thread_id for critical, single and master. To prove concept, use one shared variable for each synchronization construct.

Code:

```
#include <stdio.h>

#include <omp.h>

int main()
```

```
{
int x = 0, y = 0, z = 0;
#pragma omp parallel shared(x, y, z)
{
#pragma omp critical
{
x = x + omp_get_thread_num();
}
#pragma omp single
{
y = y + omp_get_thread_num();
}
#pragma omp master
{
z = z + omp_get_thread_num();
}
printf("Thread %d, x = %d, y = %d, z = %d\n", omp_get_thread_num(), x, y, z);
}
return 0;
}
```

Output:



```
gautam@ubuntu:~$ gcc lab_5_4.c -fopenmp
gautam@ubuntu:~$ ./a.out
Thread 0, x = 6, y = 3, z = 0
Thread 1, x = 6, y = 3, z = 0
Thread 3, x = 6, y = 3, z = 0
Thread 2, x = 6, y = 3, z = 0
```

Q5. Consider you have to write a program for VIT placement cell where 100 students are placed in 4 companies namely, Amazon, Google, Shell, and Intel. Assume no student is offered more than one placement offer. The program has to do the following tasks in parallel and display the result with thread id.

Get as input the name, register number, the pay package of students selected for jobs in the particular organization

Display the total number of students selected in each company.

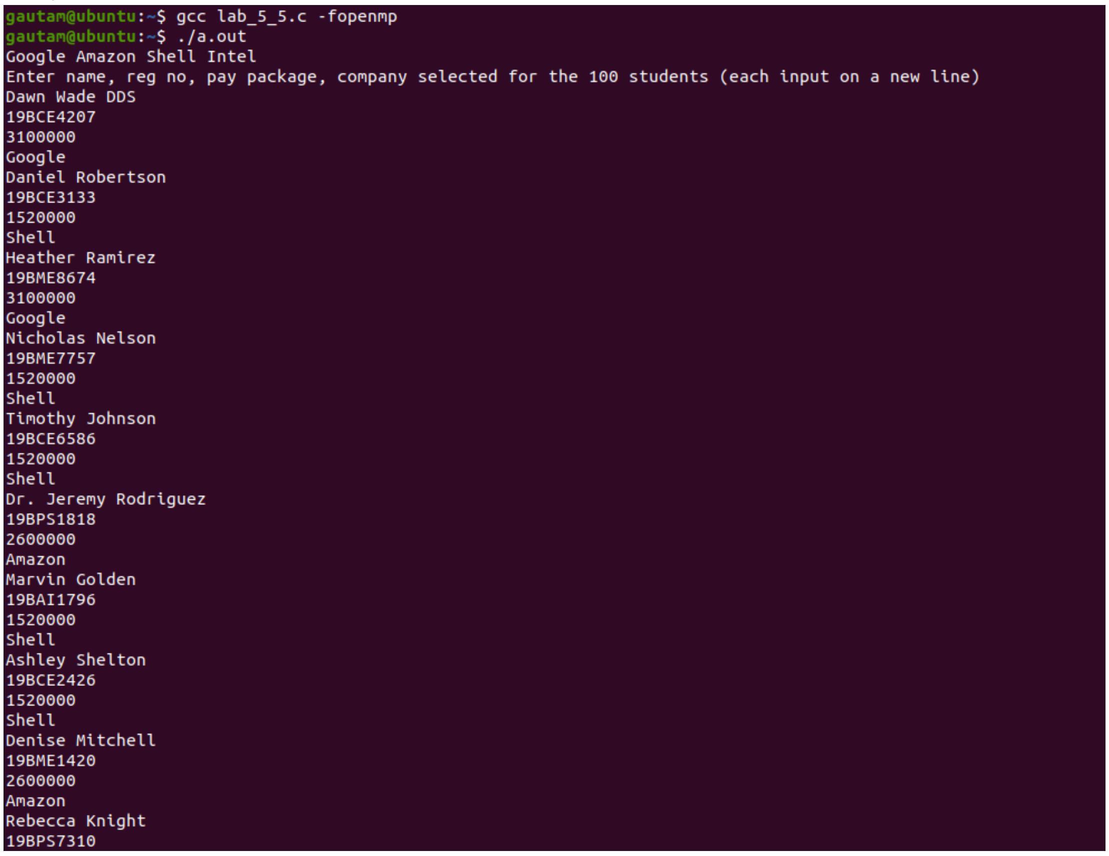
Display the average pay package of the 100 students.

Code:

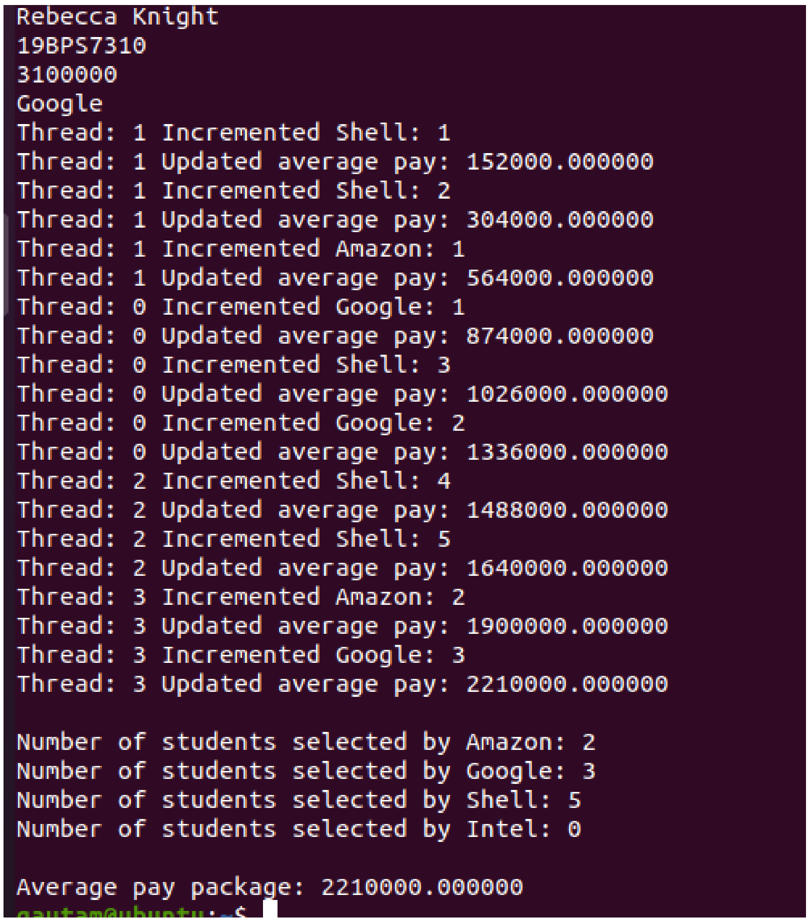
```
#include <stdio.h>

#include <omp.h>

#include<string.h>

#include<stdlib.h>


int main()

{

int n = 10;

char names[n][30];

char reg_nos[n][9];

int packages[n];

char companies[n][10];

char amazon[] = "Amazon", google[] = "Google", shell[] = "Shell", intel[] = "Intel";

printf("%s %s %s %s\n", google, amazon, shell, intel);

printf("Enter name, reg no, pay package, company selected for the 100 students (each input on a
new line)\n");

for(int i = 0; i < n; i++) {

scanf("%s", names[i]);

scanf("%s", reg_nos[i]);

scanf("%d", &packages[i]);

scanf("%s", companies[i]);

}

int num_students_amazon = 0, num_students_google = 0, num_students_shell = 0,
num_students_intel = 0;

double average_pay = 0.0f;

#pragma omp parallel for shared(packages, companies, num_students_amazon,
num_students_google, num_students_shell, num_students_intel, average_pay, amazon, google,
shell, intel)

for(int i = 0; i < n; i++)
```

```
{
#pragma omp critical
{
if(strcmp(amazon, companies[i]) == 0)
{
num_students_amazon++;
printf("Thread: %d Incremented Amazon: %d\n", omp_get_thread_num(), num_students_amazon);
}
}
#pragma omp critical
{
if(strcmp(google, companies[i]) == 0)
{
num_students_google++;
printf("Thread: %d Incremented Google: %d\n", omp_get_thread_num(), num_students_google);
}
}
#pragma omp critical
{
if(strcmp(shell, companies[i]) == 0)
{
num_students_shell++;
printf("Thread: %d Incremented Shell: %d\n", omp_get_thread_num(), num_students_shell);
}
}
#pragma omp critical
{
if(strcmp(intel, companies[i]) == 0)
{
num_students_intel++;
printf("Thread: %d Incremented Intel: %d\n", omp_get_thread_num(), num_students_intel);
}
}
#pragma omp critical
{
average_pay += 1.0f*packages[i]/n;
printf("Thread: %d Updated average pay: %f\n", omp_get_thread_num(), average_pay);
}
}
printf("\nNumber of students selected by Amazon: %d\n", num_students_amazon);
printf("Number of students selected by Google: %d\n", num_students_google);
printf("Number of students selected by Shell: %d\n", num_students_shell);
printf("Number of students selected by Intel: %d\n", num_students_intel);
printf("\nAverage pay package: %f\n", average_pay);
return 0;
}
```

Output: (For 10 students)



```
gautam@ubuntu:~$ gcc lab_5_5.c -fopenmp
gautam@ubuntu:~$ ./a.out
Google Amazon Shell Intel
Enter name, reg no, pay package, company selected for the 100 students (each input on a new line)
Dawn Wade DDS
19BCE4207
3100000
Google
Daniel Robertson
19BCE3133
1520000
Shell
Heather Ramirez
19BME8674
3100000
Google
Nicholas Nelson
19BME7757
1520000
Shell
Timothy Johnson
19BCE6586
1520000
Shell
Dr. Jeremy Rodriguez
19BPS1818
2600000
Amazon
Marvin Golden
19BAI1796
1520000
Shell
Ashley Shelton
19BCE2426
1520000
Shell
Denise Mitchell
19BME1420
2600000
Amazon
Rebecca Knight
19BPS7310
```



```
Rebecca Knight
19BPS7310
3100000
Google
Thread: 1 Incremented Shell: 1
Thread: 1 Updated average pay: 152000.000000
Thread: 1 Incremented Shell: 2
Thread: 1 Updated average pay: 304000.000000
Thread: 1 Incremented Amazon: 1
Thread: 1 Updated average pay: 564000.000000
Thread: 0 Incremented Google: 1
Thread: 0 Updated average pay: 874000.000000
Thread: 0 Incremented Shell: 3
Thread: 0 Updated average pay: 1026000.000000
Thread: 0 Incremented Google: 2
Thread: 0 Updated average pay: 1336000.000000
Thread: 2 Incremented Shell: 4
Thread: 2 Updated average pay: 1488000.000000
Thread: 2 Incremented Shell: 5
Thread: 2 Updated average pay: 1640000.000000
Thread: 3 Incremented Amazon: 2
Thread: 3 Updated average pay: 1900000.000000
Thread: 3 Incremented Google: 3
Thread: 3 Updated average pay: 2210000.000000

Number of students selected by Amazon: 2
Number of students selected by Google: 3
Number of students selected by Shell: 5
Number of students selected by Intel: 0

Average pay package: 2210000.000000
gautam@ubuntu:~$
```